Implementation of GPU accelerated SPECT reconstruction with Monte Carlo-based scatter correction

Bexelius, Tobias

2018-06

Bexelius , T & Sohlberg , A 2018 , ' Implementation of GPU accelerated SPECT reconstruction with Monte Carlo-based scatter correction ' , Annals of nuclear medicine , vol. 32 , no. 5 , pp. 337-347 . https://doi.org/10.1007/s12149-018-1252-1

http://hdl.handle.net/10138/304163
https://doi.org/10.1007/s12149-018-1252-1

publishedVersion

Annals of Nuclear Medicine (2018) 32:337–347
https://doi.org/10.1007/s12149-018-1252-1

ORIGINAL ARTICLE

Implementation of GPU accelerated SPECT reconstruction with Monte Carlo-based scatter correction

Tobias Bexelius[1] · **Antti Sohlberg**[2]

Received: 12 February 2018 / Accepted: 19 March 2018 / Published online: 21 March 2018

Abstract

Objective Statistical SPECT reconstruction can be very time-consuming especially when compensations for collimator and detector response, attenuation, and scatter are included in the reconstruction. This work proposes an accelerated SPECT reconstruction algorithm based on graphics processing unit (GPU) processing.

Methods Ordered subset expectation maximization (OSEM) algorithm with CT-based attenuation modelling, depth-dependent Gaussian convolution-based collimator-detector response modelling, and Monte Carlo-based scatter compensation was implemented using OpenCL. The OpenCL implementation was compared against the existing multi-threaded OSEM implementation running on a central processing unit (CPU) in terms of scatter-to-primary ratios, standardized uptake values (SUVs), and processing speed using mathematical phantoms and clinical multi-bed bone SPECT/CT studies.

Results The difference in scatter-to-primary ratios, visual appearance, and SUVs between GPU and CPU implementations was minor. On the other hand, at its best, the GPU implementation was noticed to be 24 times faster than the multi-threaded CPU version on a normal 128×128 matrix size 3 bed bone SPECT/CT data set when compensations for collimator and detector response, attenuation, and scatter were included.

Conclusions GPU SPECT reconstructions show great promise as an every day clinical reconstruction tool.

Keywords SPECT reconstruction · Scatter correction · Monte Carlo · Graphics processing unit (GPU)

Introduction

There are several image-related factors that affect SPECT image quality of which photon attenuation, Compton scatter, and collimator-detector response are the most detrimental. Photon attenuation can cause false defects in SPECT images and is thus likely to reduce SPECT's diagnostic accuracy [1, 2]. Compton scattering, on the other hand, lowers contrast and can cause an overestimation of the activity distribution [3]. The poor system resolution caused by the collimator and detector reduces image resolution. Fortunately, the effects of these factors can be compensated by modelling them during statistical SPECT reconstruction and there is compelling evidence, especially in nuclear cardiology, that these compensations have a significant impact on the SPECT image quality [4].

Unfortunately, these compensations can add significantly to the reconstruction time. Reconstruction times can be shortened using accelerated reconstruction algorithms [5], fast implementations [6], accelerated compensation methods [7], or using parallel processing. On the other hand, there might be a need to shift towards multi-bed studies due to the recent findings of multi-bed bone SPECT/CT superiority compared to the conventional planar imaging [8] and one major vendor is promoting the use of larger matrix in bone SPECT studies [9]. With multi-bed data sets and larger matrix sizes, it is possible that reconstructions take long time even with optimized algorithms.

One relatively new method to increase the reconstruction speed is to implement the reconstruction algorithm for the graphics processing unit (GPU) [10, 11]. GPU offers massive parallel computing capabilities at affordable prices, but it generally requires re-implementation of the algorithm written for the central processing unit (CPU), because the

✉ Antti Sohlberg
antti.sohlberg@phsotey.fi

[1] HERMES Medical Solutions, Skeppsbron 44, 111 30 Stockholm, Sweden

[2] Laboratory of Clinical Physiology and Nuclear Medicine, Joint Authority for Päijät-Häme Social and Health Care, Keskussairaalankatu 7, 15850 Lahti, Finland

CPU algorithm usually does not optimally fit the GPU programming model and architecture. This paper represents GPU implementation and validation of our ordered subsets expectation maximization (OSEM) SPECT reconstruction algorithm. The Monte Carlo (MC)-based scatter compensation method that we use presents particular challenges due to random memory access.

Materials and methods

One iteration of the OSEM algorithm generally consists of the following four steps:

1. Forward projection of the current image estimate.
2. Division of the measured projections with forward projected ones.
3. Back projection of the correction factors.
4. Update of the current image estimate with the back projected correction factors.

The following sections represent the CPU and GPU OSEM implementations of the projectors. Both implementations are part of the HybridRecon-reconstruction package (HERMES Medical Solutions, Stockholm, Sweden). The CPU implementation has been extensively validated with cardiac software phantoms and cardiac patients [7], in quantitative multi-center setup with the NEMA-IEC phantom (NEMA, Washington, District of Columbia, USA) [12], in clinical cardiac patients in half-time imaging setup [13], in Lu177 dosimetry setup [14] and against one gamma camera manufacturers own reconstruction method [15]. The CPU implementation can be considered as our "gold standard" in this work.

CPU implementation

The forward projector was implemented as rotation-based [6]. In rotation-based projectors, the projection plane is kept at fixed position and the reconstructed image is rotated (Fig. 1). This makes attenuation and collimator and detector response modelling straightforward, because attenuation correction factors for each voxel can be calculated simply by summing the rotated attenuation map along columns. Collimator and detector response modelling simplifies to convolution of each coronal plane with a response kernel, which we assumed to be depth-dependent Gaussian that does not take collimator scatter or penetration into account. Scatter modelling is performed using MC simulation and it is accelerated by simulating scatter only during the first two iterations of the OS-EM algorithm [7].

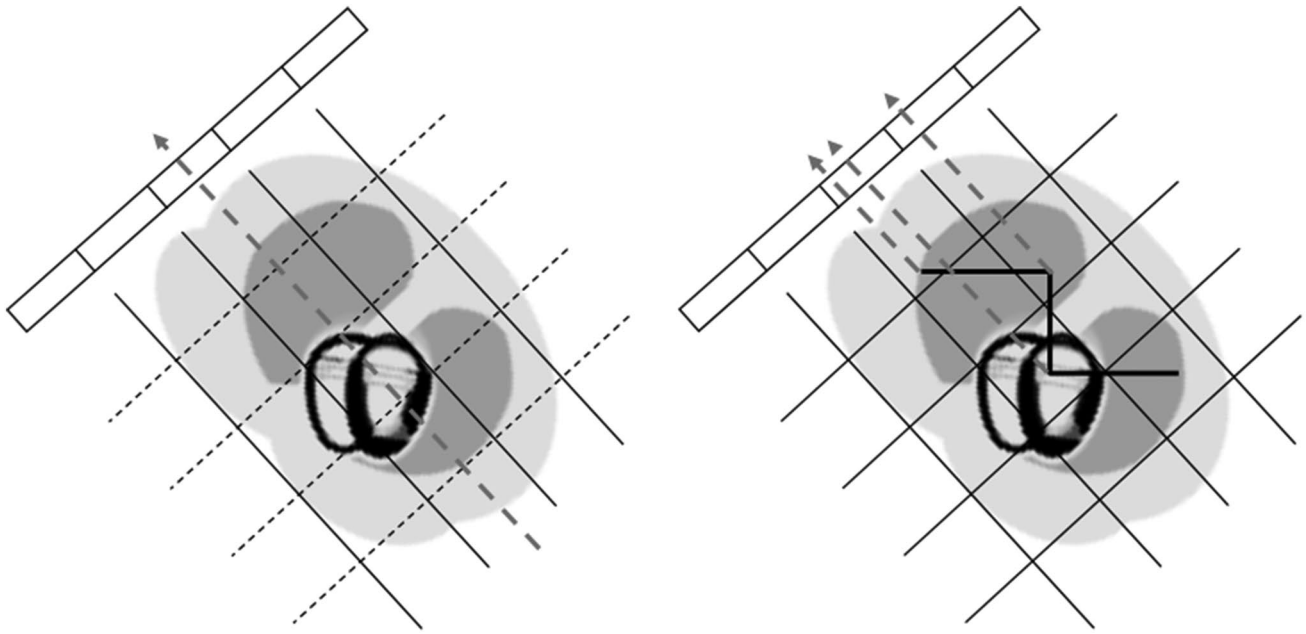

Fig. 1 Rotating projector (left) rotates the image to match the current projection angle. Projection is formed by summing along reconstruction matrix columns (dashed arrow). Attenuation model is easy to add to the column sums. Collimator and detector response can be done by convolving each coronal plane (dashed lines) with a depth-dependent response kernel. In the convolution-based forced detection (right) photon is forced to scatter (dashed arrows) towards the detector at each scattering site. The current 3D subprojection map (the overlaid grid) is updated by the photon weight. After all photons have been traced, projection is formed in identical manner as with the rotating projector

The MC scatter simulation is initialized by distributing photons to be traced to voxels according to the emission map voxel values. Voxel i has a probability of c_i/c_t, where c_i is the number of counts in the voxel and c_t the total number of counts in the emission map, to emit a photon. Photons are randomly sampled from the distribution; and then, the energy, direction, and distance to next interaction location are sampled and the photon's weight is initialized to 1. Energy is sampled from current isotope's energy spectrum, direction is uniformly sampled from unit sphere, and distance is calculated based on the Woodcock tracking algorithm [16] as $-\ln(r)/\mu_{MAX}$, where r is a random number in range [0, 1] and μ_{MAX} is the maximum linear attenuation coefficient in the attenuation map. Attenuation map is generated from the density map by multiplying it with tissue's energy-dependent mass attenuation coefficient obtained from cross-section tables defined by Berger and Hubbell [17]. At each interaction site, a random number is drawn and compared to the probability proportional to μ_i/μ_t, where μ_i is the current voxel's linear attenuation coefficient and μ_t the maximum attenuation coefficient of the volume. If the random number is smaller than the attenuation coefficient ratio, forced non-attenuation is performed (photon's weight is multiplied by probability of not going through photo-electric absorption). The photon is also forced to scatter towards the detector according to the convolution-based forced detection principle [18] and a 3D subprojection map is updated by the photon's weight (Fig. 1). In addition to forced detection, the photon will also go through Compton scattering where its direction and energy is updated. In case of a random number larger than the ratio, the photon continues along its original direction. This process repeats itself until the photon has scattered a predetermined number of times (in our case 10), its energy drops under predetermined limit (in our case under 75% of the acquisition energy window lower limit) or until it leaves the density map. After all photons have been simulated, the 3D subprojection map is forward projected using the above-mentioned projector to create scatter projections.

The back projector was also implemented as rotation-based. Back projection is performed by accumulating the correction factors for each reconstruction voxel from the projection plane using attenuation and collimator and detector response modelling and then rotating the back projected image to its correct angle. We omitted scatter modelling from back projection to increase speed.

The CPU algorithm was parallelized by computing forward projection and back projection for each projection angle on a separate processor.

GPU implementation

The current CPU implementation needed finer detail multithreading to fully utilize the power of the GPU, and thus, the CPU projectors were rewritten.

The forward projector was implemented as ray-driven, i.e., projection rays are cast from the center of each projection pixel and the forward projector is launched as an OpenCL kernel with one thread per projection pixel (Fig. 2). OpenCL was used, because it does not limit the implementation to NVIDIA graphics cards like CUDA. The current image estimate is stored as a 3D image on the GPU. The projection pixel value is calculated by stepping the ray one voxel length at a time and image voxel values and attenuation coefficients obtained from an aligned attenuation map are interpolated using specialized hardware on the GPU. Collimator and detector response modelling is performed by assuming the response to be a 2D depth-dependent Gaussian function as in the CPU implementation. However, instead of convolving entire coronal planes with 2D Gaussians, the projection ray and reconstruction voxel intersection neighborhood is sampled using a 2D Gauss kernel. The kernels are precalculated and stored on the GPU.

MC-based scatter modelling is difficult to implement efficiently on GPUs for two main reasons: divergence of threads and generation of pseudorandom numbers. Most pseudorandom number generators work in a sequential manner, using a seed as input and returning a new seed

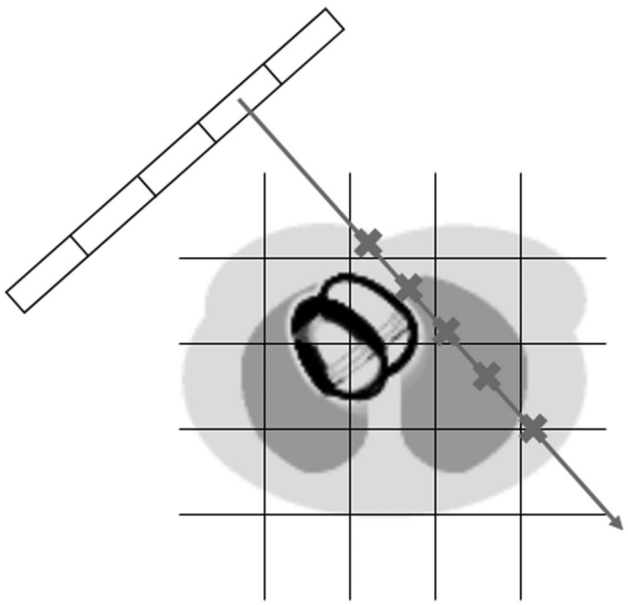

Fig. 2 Ray-driven projector casts a ray from the center of each projection pixel. Ray value is accumulated by summing along the projection ray. The voxel values are sampled at the "x"-locations using a 2D Gaussian function whose width corresponds to the resolution of the collimator–detector

for each generated random number. To parallelize the MC simulation, a random number generator was needed that could generate random numbers in thousands of threads independently, with no statistical correlation between each other. The counter-based Philox-4 × 32_10 algorithm [19] was used as the random number generator and we used the technique in [20] to convert the integer random numbers to floating point values. Since random values are used for directing each ray through the volume, they will quickly diverge which would make it impossible for threads to update the 3D subprojection map in parallel without data races, since multiple workgroups cannot synchronize their access to global GPU memory. To mitigate these problems, we need to continuously sort the rays as they scatter through the volume.

The GPU simulates thousands of photons in parallel, and can only synchronize small groups of them at once (except for simple counting), which is why we start by splitting the emission map in blocks of 4 × 4 × 8 voxels. The photons are then distributed by first sampling which block each simulated photon will belong to, and after that by randomly distributing all photons within each block.

The photons are traced in parallel using same method as on CPU, using the Woodcock algorithm [16], until each photon has found their next interaction site or left the volume. Before doing forced detection, which requires us to synchronize the update of the 3D subprojection map between all detected photons, we need to sort the photons based on voxel index. Only a few sorting algorithms may be efficiently used on GPU, and we have chosen radix sort which is the current state-of-the-art for sorting on GPU [21]. We run the radix sort iteratively k times in groups of 4 bits each from least to most significant bit, except for last 4–7 bits. This will effectively group the photons in 2^{4k} neighborhoods of 16–128 voxels each, which now allows us to perform forced detection in a synchronized manner for all photons in each such neighborhood. The sorting algorithm is also where we remove photons that have either left the volume during tracing, or whose energy has dropped to under the predetermined level.

When doing the forced detection in the CPU forward projector, the limited energy resolution of the detector is taken into account by sampling the detected photon energy from a Gaussian distribution with a width corresponding to the energy resolution of the detector. This sampling was done using the Box–Muller transformation, which has to iterate until a certain condition is met and does not lend itself well to GPUs. Instead of using the Box–Muller transformation on GPU, we analytically calculate the probability of a photon with certain energy to be detected in the current energy window on a detector with certain energy resolution. This requires us to calculate the cumulative distribution function of the Gaussian energy resolution function, using the method suggested by [22].

We continue to iterate the whole scattering process a predetermined number of times (in our case 10, just as on CPU), each time checking how many photons were still alive at last iteration to avoid doing unnecessary work. When all iterations are done, the 3D subprojection map is forward projected using the ray-driven projector explained above.

Just as for forward projection, back projection was implemented as ray-driven and launched as an OpenCL kernel with one thread per projection pixel on the projection plane, stepping through the reconstruction volume starting from the projection plane. Back projection takes the same steps through the volume as the forward projection and can thus reuse the same precalculated Gauss kernels which were generated prior to forward projection. As the back projector iterates through the reconstructed volume, each thread uses the precalculated Gauss kernels to sample the prefetched input projection from local GPU memory, and attenuates it using the thread's accumulated density, and writes the resulting value for each step to a temporary volume that is aligned with the current projection angle, so that each thread touches its own column of the volume. This volume is stored as a buffer in the GPU global memory. After the threads have stepped through and filled the volume in the back projection OpenCL kernel, the resulting buffer is copied into a temporary 3D image, which is rotated with hardware-based linear interpolation and added to the final set of correction factor, which are used to update the image.

XCAT phantom study

The already validated CPU and newly developed GPU reconstruction algorithms were compared with mathematical phantoms and patient images. The XCAT phantom [23] was used to compare the CPU and GPU forward projectors. In the first phantom study a Tc99m point source was placed in the XCAT phantom's density map according to Fig. 3. The following densities were used for lung, soft tissue, and bone: 0.49, 1.0, and 1.5 g/cm^3. In the second study, the XCAT phantom was used to model Tc99m-HDP bone uptake with the following relative activities for bone, kidneys, and soft tissue: 700, 250, and 40 (Fig. 3). The CPU and GPU projectors were used to generate anterior projections of the two XCAT phantom setups. Siemens Symbia T gamma camera detector (3/8″ NaI crystal, 3.9 mm intrinsic resolution and 9.9% energy resolution) and LEHR collimator (2.405 cm hole length and 0.111 cm hole diameter) were used in the simulations with 128 × 128 matrix size, 5.0 mm pixel size, and 250 mm radius of rotation. The number of simulated photons was set to 1,015,808 due to the GPU implementation, which groups voxels into 4 × 4 × 8 voxel blocks and thus exactly 1,000,000 could not be used. The projectors were

Fig. 3 Transverse slice of XCAT density map (left) and emission map (right) used to validate the CPU and GPU forward projectors. Point source location is marked by an arrow

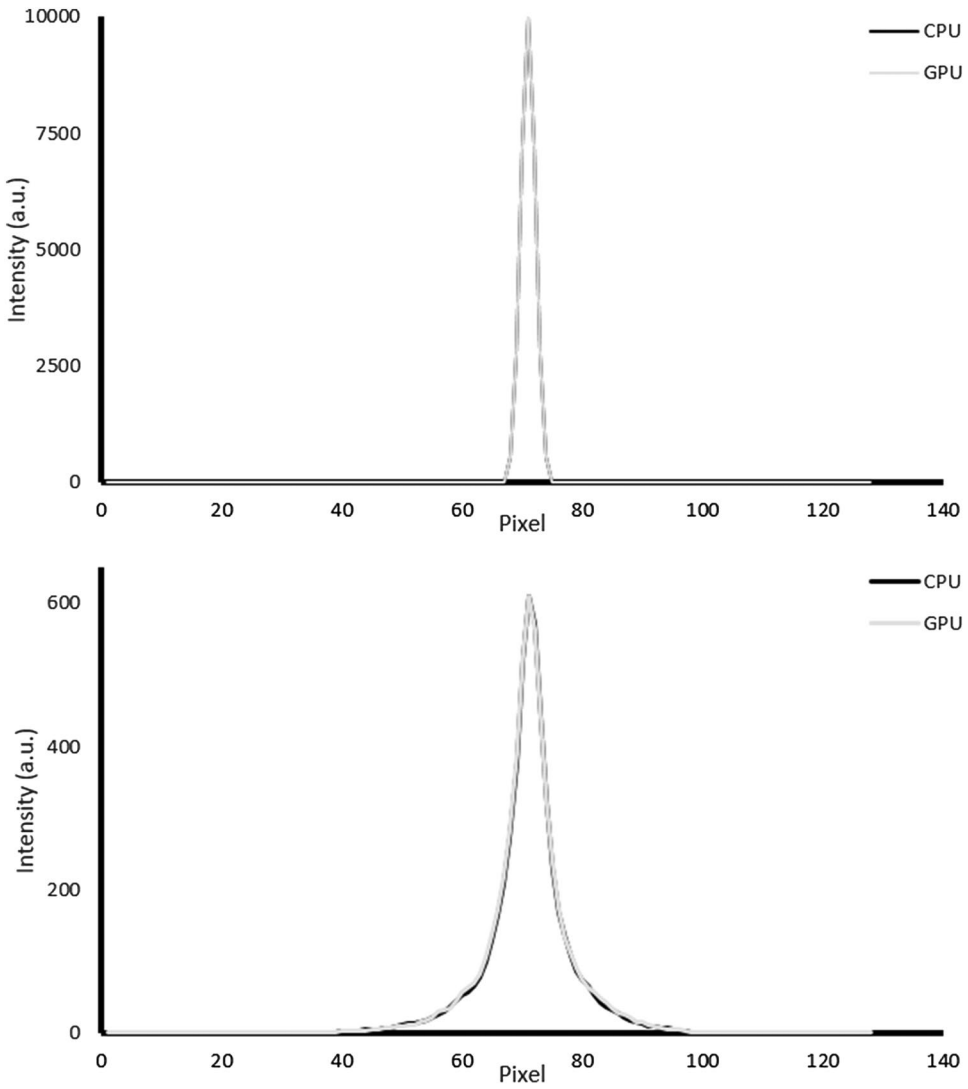

Fig. 4 Profiles through CPU and GPU primary (top) and scatter (bottom) projections of the XCAT point source study

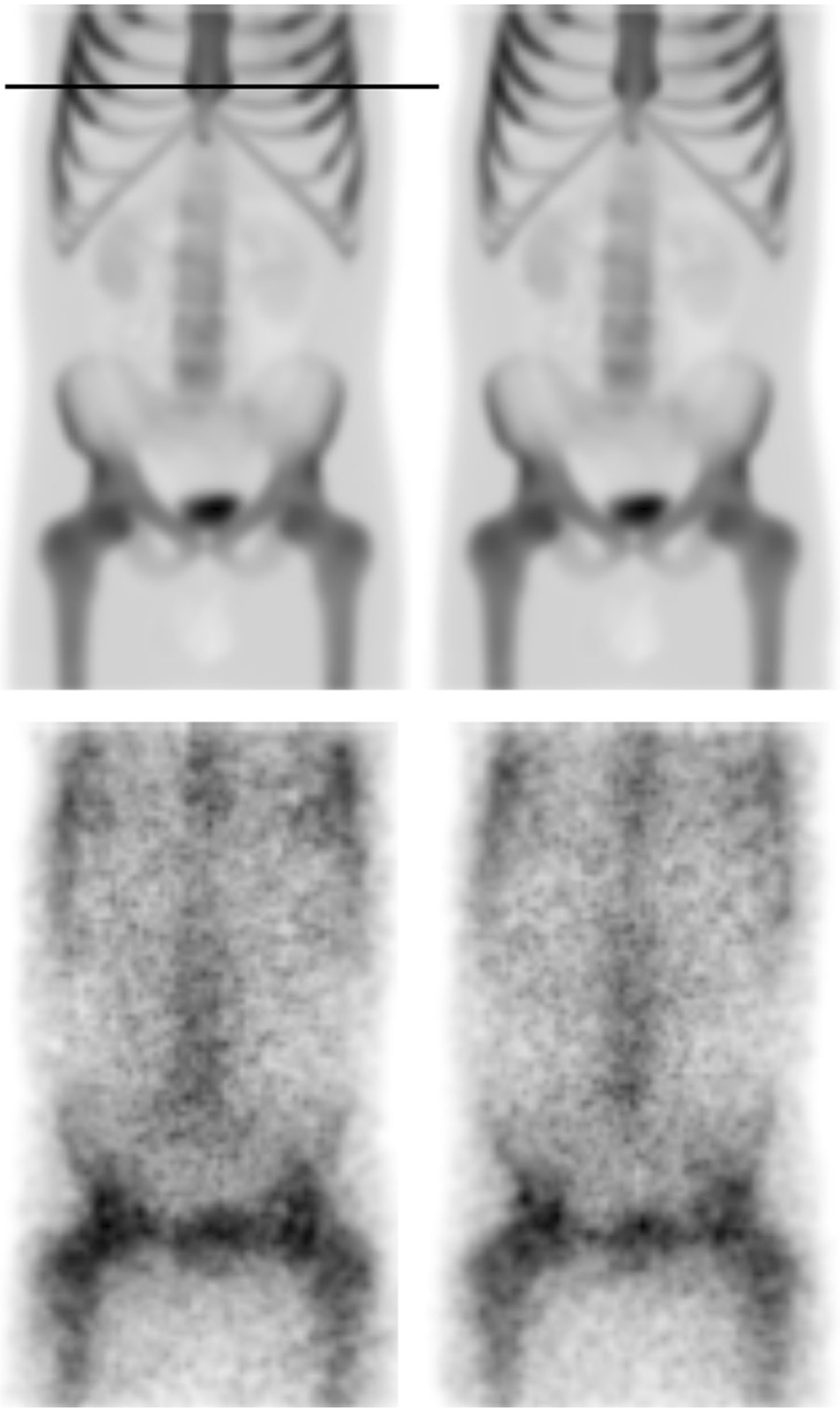

Fig. 5 XCAT bone study primary (top) and scatter (bottom) projections generated with CPU (left) and GPU (right). The horizontal black line shows the location of the profiles shown in Fig. 6

compared by calculating the scatter-to-primary count ratios of the projections and by visually analyzing the profiles taken over the projections of primary and scattered counts.

Patient studies

Five three bed bone SPECT/CT studies were performed

with Tc99m-HDP using a Siemens Symbia T and reconstructed with the CPU and GPU algorithms. Studies were acquired using 128 × 128 matrix size, 4.8 mm pixel size, 64 projection angles over 360 degree rotation, and body contour orbit. Reconstructions were run using CPU and GPU OSEM algorithms with 16 subsets and 5 iterations, and with attenuation, collimator response, and scatter compensations. The MC-based scatter compensation was used with 122,880 simulated photons per projection due to GPU implementation, which did not allow exactly 100,000 photons to be used, and simulating scatter only during the first 2 OSEM iterations. The reconstruction methods were analyzed by comparing mean SUVs (10 mm diameter spherical VOI) at ten different locations within each patient randomly extracted from the spine area. VOIs were drawn in HybridViewer (HERMES Medical Solutions, Stockholm, Sweden).

Results

XCAT phantom study

The scatter-to-primary ratio for the XCAT point source study reprojected with CPU was 0.856 and with GPU 0.859, whereas the corresponding values were 0.343 and 0.341 for the bone study. Figure 4 shows horizontal profiles through the primary and scatter projections of the point source study. The primary projection profiles overlap completely and scatter projection profiles also look very similar. Figure 5 shows CPU and GPU primary and scatter projections of the XCAT bone study. Visually, the projections look identical. As in the point source study, the primary projection profiles overlap completely and scatter projection profiles look similar.

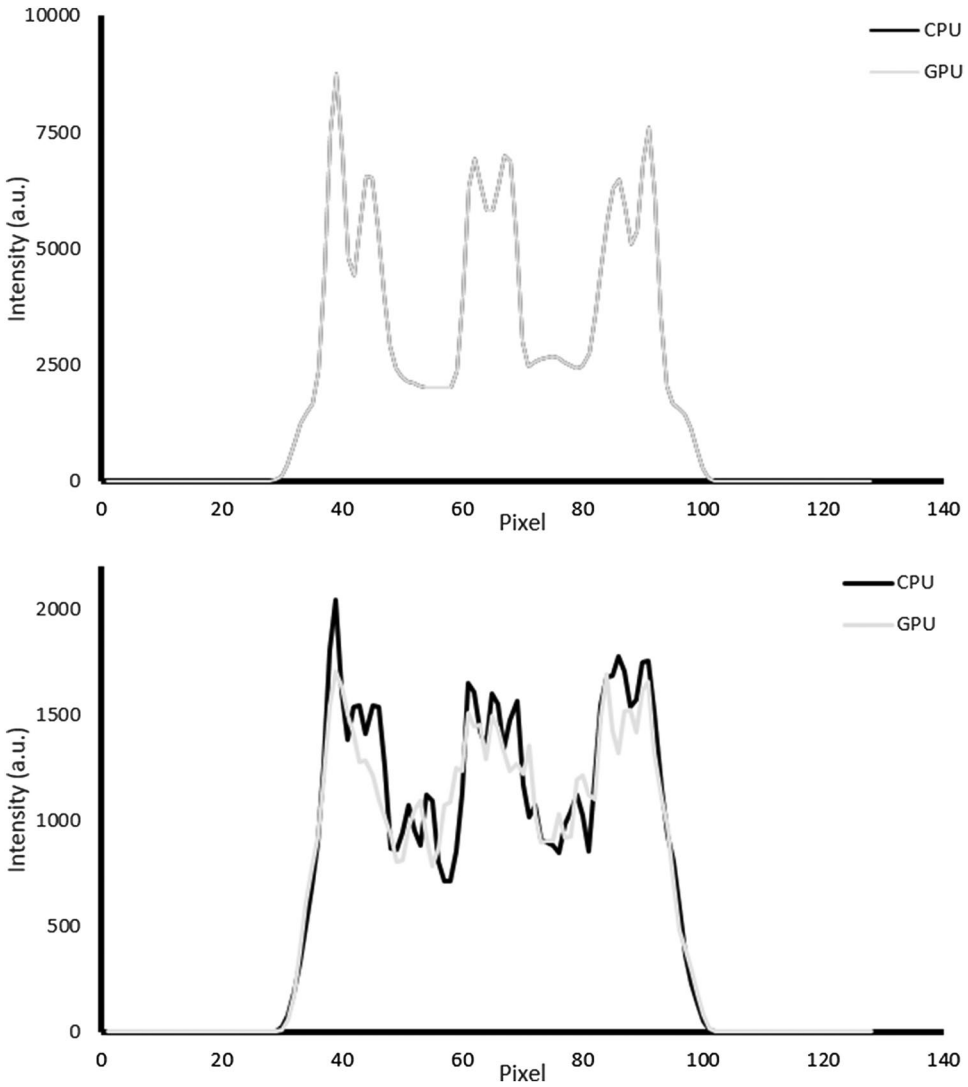

Fig. 6 Profiles through CPU and GPU primary (top) and scatter (bottom) projections of the XCAT bone study

Patient studies

The SUVs of the CPU and GPU reconstruction are compared in Fig. 7. The SUVs were well correlated (correlation coefficient = 0.91), but the GPU SUVs were slightly higher (average difference 0.25) than the CPU SUVs. This difference was noticed to be statistically significant (p value < 0.05). The Bland–Altman plot is shown in Fig. 8. The lower and upper limits of agreement were − 1.24 and 0.74. Example maximum intensity projection (MIP) images reconstructed with CPU and GPU are shown in Fig. 9, and visually, they appear to be identical. Average reconstruction times of the patient studies are shown in Table 1 and the effect of the attenuation, collimator response and scatter compensation on the reconstruction time is shown in Fig. 10.

Discussion

The reconstruction times of SPECT images using state-of-the art compensation methods can be long. In this paper, we presented a GPU implementation of an OSEM-based SPECT reconstruction algorithm with attenuation, collimator response and MC-based scatter compensation. The implementation was validated using phantom and patient studies. We had to completely rewrite the projectors of the OSEM algorithm in order to fully utilize GPUs computing

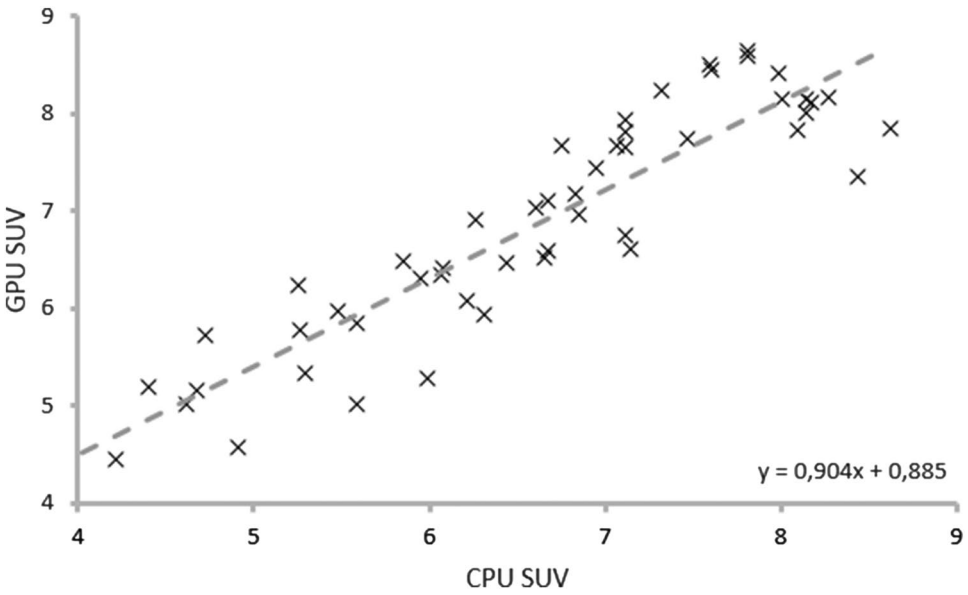

Fig. 7 Correlation between CPU and GPU SUVs. The dashed line presents the best fit

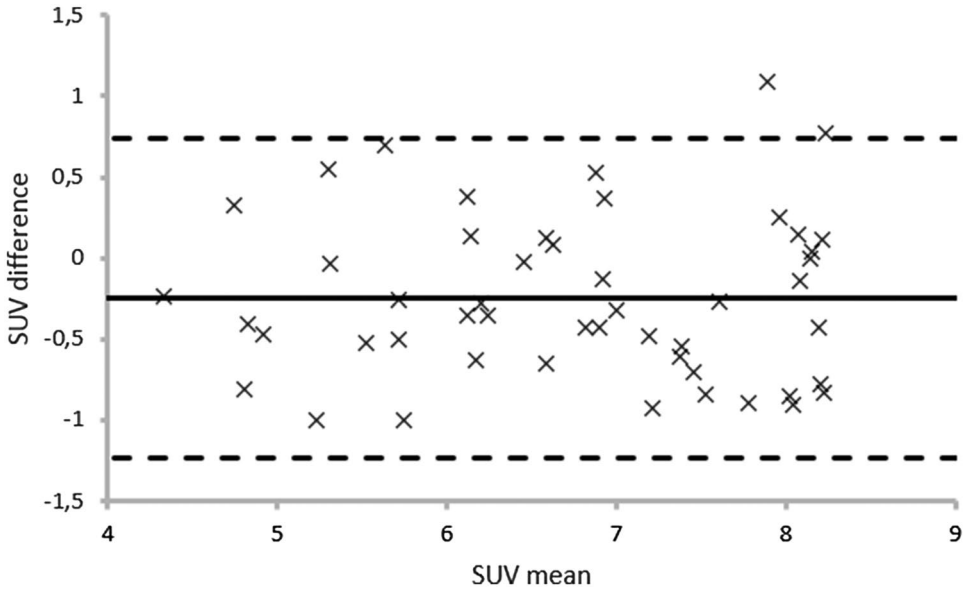

Fig. 8 Bland–Altman plot of CPU and GPU SUVs. The mean difference (average of CPU SUV–GPU SUV) is shown as solid line and the limits of agreement (− 1.96 standard deviation and + 1.96 standard deviation) are shown as dashed lines

power, and thus, the results of the CPU and GPU reconstruction algorithms were not expected to be identical.

Despite the rewrite, the CPU and GPU forward projections match well (Figs. 4, 5, 6) and the scatter-to-primary ratios are close to each other. In patient studies, GPU SUVs were noticed to be slightly, but statistically significantly, higher than the CPU SUVs (Figs. 7, 8). This is probably due to GPU algorithm's ray-driven projector, which does not include extra interpolation like our rotating projector used in the CPU implementation, leading to better resolution. Visually (Fig. 9), it is, however, impossible to separate images reconstructed with GPU from CPU images.

The reconstruction time is heavily dependent of the type of the graphics card, as shown in Table 1 and Fig. 10. The lowest speedup factor is achieved with the K-card, which is the oldest of the tested cards. The newer M-card is approximately twice as fast as the K-card and the latest Geforce is more than 10 times faster than the K5000. Interestingly K5000 performs worse than CPU in MC-based scatter compensation (K5000 reconstruction time increases 39.9 s when scatter compensation is added to attenuation and collimator response compensation, whereas, with CPU, the time increases only by 34 s, see Fig. 10). MC-based scatter compensation is also the most demanding part on the other graphics cards too. The better performance of the newer cards is based on improved architecture, more cores, higher clock rate, and higher memory bandwidth.

Table 1 Average reconstruction times (average of all 5 patients and 5 individual reconstruction algorithm runs per patient) and speedup factors relative to CPU for CPU (4 cores on Xeon E5-1650 v4 @3.60 GHz) and three different type of graphics cards

	CPU	Quadro K5000	Quadro M2000	Geforce GTX 1080Ti
Time	91.8 s	43.6 s	17.9 s	3.8 s
Speedup factor	× 1	× 2	× 5	× 24

GPU reconstruction has been studied more in PET [11, 24, 25] than in SPECT. It is difficult to directly compare the speedup factors obtained with GPU, because they depend on the CPU implementation. In our case, CPU code is already multi-threaded and optimized, and thus, our GPU speedup factors are lower than for example by

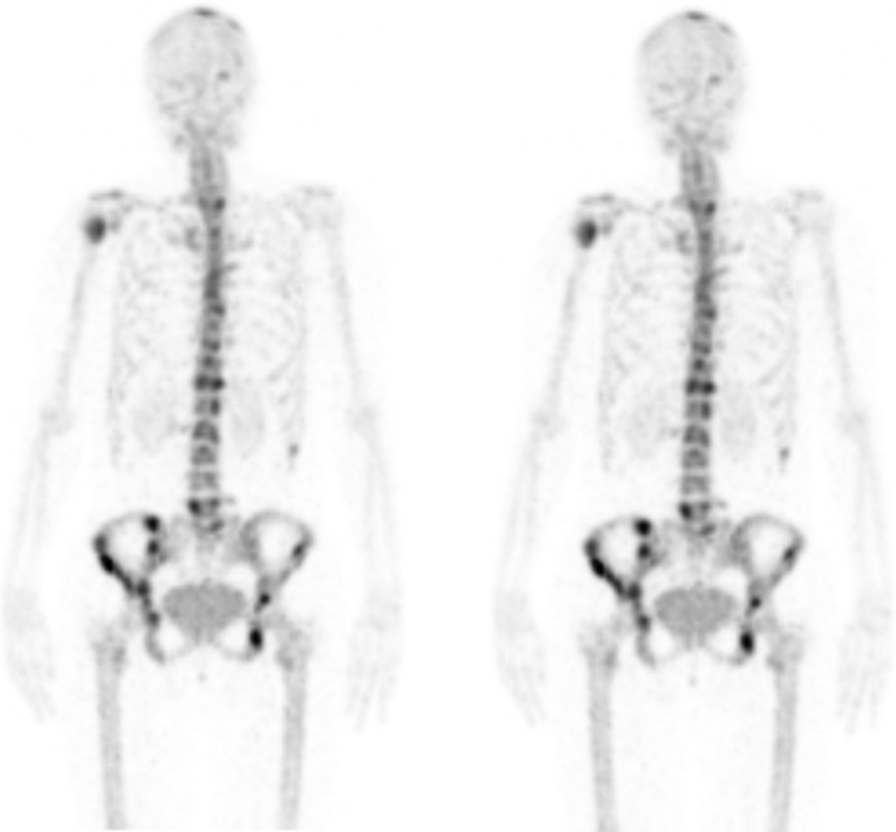

Fig. 9 Anterior MIP projections obtained with CPU (left) and GPU (right)

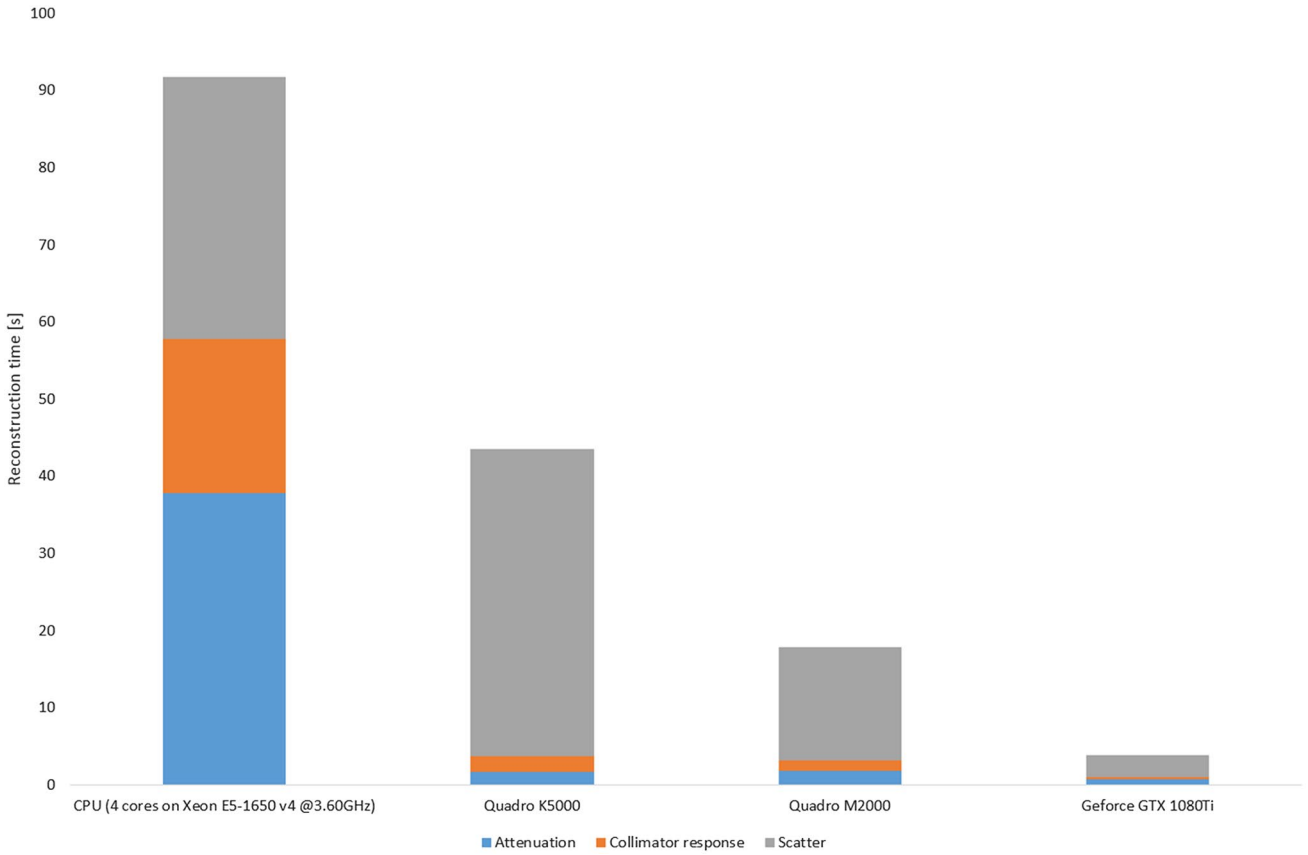

Fig. 10 Time spend on attenuation, collimator response, and scatter compensation on CPU and the three graphics cards. The total height of the bar corresponds to the total reconstruction time

Pedemonde et al., who reported speedup factor of 75 for 128×128 matrix size [10]. On the other hand, our GPU reconstruction times are close to theirs indicating successful implementation.

In conclusion, SPECT reconstruction with MC-based scatter compensation can be accelerated with GPU, so that multi-bed reconstructions can be run in less than 4 s approaching real-time processing.

Acknowledgements Tobias Bexelius works for HERMES Medical Solutions and Antti Sohlberg has a consulting agreement with HERMES Medical Solutions.

References

1. Narayanan MV, King MA, Pretorius PH, Dahlberg ST, Spencer F, Simon E, et al. Human-observer receiver-operating-characteristic evaluation of attenuation, scatter, and resolution compensation strategies for (99 m)Tc myocardial perfusion imaging. J Nucl Med. 2003;44:1725–34.
2. Niu X, Yang Y, Jin M, Wernick MN, King MA. Effects of motion, attenuation, and scatter corrections on gated cardiac SPECT reconstruction. Med Phys. 2011;38:6571–84.
3. Zaidi H, Koral KF. Scatter modelling and compensation in emission tomography. Eur J Nucl Med Mol Imaging. 2004;31:761 – 82.
4. Frey EC, Gilland KL, Tsui BM. Application of task-based measures of image quality to optimization and evaluation of three-dimensional reconstruction-based compensation methods in myocardial perfusion SPECT. IEEE Trans Med Imaging. 2002;21:1040–50.
5. Hudson HM, Larkin RS. Accelerated image reconstruction using ordered subsets of projection data. IEEE Trans Med Imag. 1994;13:601–9.
6. Di Bella EVR, Barclay AB, Eisner RL, Schafer RW. A comparison of rotation-based methods for iterative reconstruction algorithms. IEEE Trans Nucl Sci. 1996;43:3370–6.
7. Sohlberg A, Watabe H, Iida H. Acceleration of Monte Carlo-based scatter compensation for cardiac SPECT. Phys Med Biol. 2008;53:N277–N285.
8. Jambor I, Kuisma A, Ramadan S, Huovinen R, Sandell M, Kajander S. et. al. Prospective evaluation of planar bone scintigraphy, SPECT, SPECT/CT, 18F-NaF PET/CT and whole body 1.5T MRI, including DWI, for the detection of bone metastases in high risk breast and prostate cancer patients: SKELETA clinical trial. Acta Oncol. 2016;55:59–67.
9. Delcroix O, Robin P, Gouillou M, Le Duc-Pennec A, Alavi Z, Le Roux PY, et al. New SPECT/CT reconstruction algorithm: reliability and accuracy in clinical routine for non-oncologic bone diseases. EJNMMI Res. 2018;8:14.

10. Pedemonte S, Bousse A, Erlandsson K, Modat M, Arridge S, Hutton BF, et al. GPU accelerated rotation-based emission tomography reconstruction. IEEE Nuclear Science Symposium Conference Record 2010;2657–2661.
11. Ha S, Matej S, Ispiryan M, Mueller K. GPU-accelerated forward and back-projections with spatially varying kernels for 3D DIRECT TOF PET reconstruction. IEEE Trans Nucl Sci. 2013;60:166–73.
12. Kangasmaa TS, Constable C, Hippeläinen E, Sohlberg AO. Multicenter evaluation of single-photon emission computed tomography quantification with third-party reconstruction software. Nucl Med Commun. 2016;37:983–7.
13. Kangasmaa TS, Kuikka JT, Vanninen EJ, Mussalo HM, Laitinen TP, Sohlberg AO. Half-time myocardial perfusion SPECT imaging with attenuation and Monte Carlo-based scatter correction. Nucl Med Commun. 2011;32:1040–5.
14. Hippeläinen E, Tenhunen M, Mäenpää H, Sohlberg A. Quantitative accuracy of Lu-177 SPECT reconstruction using different compensation methods: phantom and patient studies. EJNMMI Res. 2016;6:16.
15. Woliner-van der Weg W, Deden LN, Meeuwis AP, Koenrades M, Peeters LH, Kuipers H, Laanstra GJ, Gotthardt M, Slump CH, Visser EP. A 3D-printed anatomical pancreas and kidney phantom for optimizing SPECT/CT reconstruction settings in beta cell imaging using 111In-exendin. EJNMMI Phys. 2016;3:29.
16. Woodcock E, Murphy T, Hemmings P, Longworth S. Techniques used in the GEM code for Monte Carlo neutronics calculations in reactors and other systems of complex geometry. Proc Conf Appl Comput Methods Reactor Probl. 1965;557:2.
17. Berger M, Hubbell J. XCOM. Photon cross sections on a personal computer. Natl Bur Stand Washington, DC (USA). Cent Radiat Res. 1987.
18. De Jong HWAM., Slijpen ETP, Beekman FJ. Acceleration of Monte Carlo SPECT simulation using convolution-based forced detection. IEEE Trans Nucl Sci. 2001;48:58–64.
19. Salmon JK, Moraes MA, Dror RO, Shaw DE. Parallel random numbers: as easy as 1, 2, 3. In: Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis 2011;1–12.
20. Doornik JA. Conversion of high-period random numbers to floating point. ACM Trans Model Comput Simul. 2007;17:3.
21. Satish N, Harris M, Garland M. Designing efficient sorting algorithms for manycore GPUs. In: Proceedings of the 2009 IEEE International Symposium on Parallel and Distributed Processing. 2009. pp. 1–10.
22. Zelen M, Severo NC. Probability functions. Handb Math Funct 1964 5;925–995.
23. Segars WP, Sturgeon G, Mendonca S, Grimes J, Tsui BMW. 4D XCAT phantom for multimodality imaging research. Med Phys. 2010;37:4902–15.
24. Pratx G, Chinn G, Olcott PD, Levin CS. Fast, accurate and shift-varying line projections for iterative reconstruction using the GPU. IEEE Trans Med Imag 2009;28:435–445.
25. Despres P, Jia X. A review of GPU-based medical image reconstruction. Phys Med. 2017;42:76–92.